Facultad de Ciencias

Sistema de gestión de los elementos y situaciones vinculados a un hecho dentro del ámbito de una ciudad inteligente

(System of management of the elements and situations linked to an event within the scope of a smart city)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Daniel Peñil Núñez

Director: Rafael Menéndez de Llano Rozas

Co-Director: José Miguel Prellezo Gutiérrez

Febrero – 2020

ÍNDICE GENERAL

Índice de ilustraciones

Lista de acrónimos

- AJAX – Asynchronous JavaScript and XML
- API – Application Programming Interface
- CPU – Central Processing Unit
- CSS – Cascading Style Sheet
- ETL – Extract, Transform and Load
- HTML – HyperText Markup Language
- JSON – JavaScript Object Notation
- MVC – Model View Controller
- NoSQL – Not Only SQL
- NGSI – Next Generation Service Interface
- RAM – Random Access Memory
- SFTP – Secure File Transfer Protocol
- SQL – Structured Query Language
- SSH – Secure Shell
- TB - TeraByte

AGRADECIMIENTOS

Con la realización de este Trabajo Fin de Grado doy por concluida mi etapa académica y no se me ocurre mejor momento que este para dedicar unas líneas de agradecimiento a todas aquellas personas que me han ayudado a llegar hasta aquí.

En primer lugar, me gustaría agradecer a mi familia el apoyo y la libertad de decisión que me han ofrecido sobre mis estudios y que me han permitido enfocar mi vida hacia una profesión que me llena y me hace feliz. En segundo lugar, quisiera dar las gracias a todas aquellas personas que han dedicado alguna parte de su tiempo, por mínima que haya sido, para transmitirme sus conocimientos sin haber puesto nunca una mala cara, en este grupo incluyo tanto a compañeros de estudios y de trabajo, así como a los numerosos profesores que he conocido durante estos años. También me gustaría agradecer a mis amigos por animarme siempre a hacer lo que me gusta y nunca rendirme, además de por todos esos momentos en los que se han mostrado sorprendidos por el trabajo que desempeño, lo cual me llena de satisfacción.

En último lugar, y no por ello menos importante, voy a estar eternamente agradecido a CIC Consulting Informático por la multitud de oportunidades que me vienen brindando durante estos últimos años y que me permiten cada día crecer como persona y como profesional.

RESUMEN

Gracias a los grandes avances en las tecnologías y las comunicaciones en los últimos años se ha popularizado el término *big data* relacionado con la ingesta de grandes cantidades de datos y su posterior tratamiento para la obtención de información útil. A raíz de este término surge el concepto de ciudad inteligente, el cual se refiere a la utilización de esta información en el ámbito de las ciudades para tomar mejores decisiones y poder aumentar la calidad de vida de la población.

Dada la necesidad de acceder y manejar dicha información para alcanzar la finalidad de convertir las ciudades en inteligentes, ha surgido el desarrollo de sistemas que llevan a cabo el proceso completo de informatización de las ciudades, desde la recogida de datos, hasta la puesta a disposición de la información a funcionarios y ciudadanos a través de aplicaciones web.

El objetivo de este proyecto es el de implementar un módulo en una aplicación web, a través del cual se pueda mejorar la toma de decisiones en momentos clave de eventos e incidentes que suceden en la ciudad, a través del conocimiento de la situación en tiempo real de los elementos cercanos a estos.

Palabras clave: ciudad inteligente, big data, Node.js, JavaScript, NoSQL, modelo vista controlador (MVC), AngularJS.

ABSTRACT

Thanks to the great advances in technologies and communications in recent years, the term big data has been popularized in relation to the intake of large amounts of data and the subsequent treatment to obtain useful information. Following on from this term the concept of smart city arises, which refers to the use of this information in the context of cities to make better decisions and to increase the quality of life of the population.

Given the need to access and manage such information to achieve the aim of converting cities into smart, the development of systems that carry out the complete process of computerization of cities has arisen, from the collection of data to the availability of the information to officials and citizens through web applications.

The objective of this project is to implement a module in a web application in this area, through which decision making can be improved at key moments of events and incidents that take place in the city, through knowledge of the situation in real time of the nearby elements.

Palabras clave: smart city, big data, Node.js, JavaScript, NoSQL, model view controller (MVC), AngularJS.

1. Introducción

Este primer apartado presenta al lector, de forma breve, el contexto en el que surge la necesidad de llevar a cabo este proyecto. Además, se describen las motivaciones y los objetivos de éste y se detalla la estructuración de la presente memoria.

1.1. Introducción

Podría considerarse que la definición que más se acerca al contexto de este proyecto es la proporcionada por C. Harrison[1] que denota una ciudad inteligente como aquella "instrumentada, interconectada e inteligente". En primer lugar, menciona que debe ser una ciudad instrumentada, esto quiere decir que la ciudad debe poseer una infraestructura dedicada a capturar e integrar datos en tiempo real a través de todo tipo de entradas, desde sensores de temperatura o cámaras, hasta incluso la información que se comparte en las redes sociales dentro de ella. A continuación, se indica que la ciudad ha de encontrarse interconectada, lo que se traduce en que debe poseer sistemas capaces de integrar los datos recogidos a través de procesos que unen dichos datos con otros de relevancia para redirigirlos a los diferentes sistemas a los que puedan resultar de interés. Por último, y quizás la parte esencial, se deben procesar los datos recibidos en los diferentes sistemas para obtener información útil que ayude optimizar la toma de decisiones en todo tipo de aspectos de la ciudad.

Dicho concepto viene fuertemente ligado a otro que se podría decir que está teniendo un impacto incluso mayor en la sociedad en la última década, que es el de *big data*. El término *big data*[2] hace referencia al análisis de extensos volúmenes de datos provenientes de todo tipo de fuentes para obtener información útil de ellos. Para entender mejor el concepto, se define a través de "las tres V": volumen, se trabaja con una inmensa cantidad de muestras de datos no estructurados; velocidad, los datos son recogidos a una gran velocidad y muchas veces son directamente procesados en tiempo real; variedad, los datos ya no presentan los tipos estructurados de unas décadas atrás y pueden

presentarse en otros formatos, como pueden ser audios o vídeos, y requerir un preprocesamiento diferente para obtener información de ellos. Hoy en día, aunque mucha gente lo desconozca, el *big data* está presente en numerosos sectores, por ejemplo, las grandes compañías analizan grandes cantidades de datos sobre las personas y sus gustos para ofrecer productos y servicios que atraigan a la mayor cantidad de consumidores. Otra de las aplicaciones que se nutre en gran medida de la aparición de este concepto es la del *machine learning*[26], la cual hace referencia al proceso de preparación de datos para llevar a cabo el entrenamiento de un algoritmo para posteriormente poder obtener predicciones ante situaciones que aún no se han dado gracias a la similitud con otros casos y al conocimiento adquirido.

1.2. Motivación y objetivos

Dentro del marco indicado en el apartado anterior, se presenta la necesidad de aprovechar la información que se puede extraer de los grandes volúmenes de datos de los que se dispone y mostrársela a diferentes grupos de usuarios para ayudarles en la toma de decisiones y la monitorización de numerosos aspectos de la ciudad. Ante esta necesidad, CIC Consulting Informático decide llevar a cabo el desarrollo de un producto que englobe, desde la recogida de datos de diferentes fuentes, hasta la interfaz de usuario que proporcione a cada uno de estos grupos de usuarios la capacidad de aprovechar la información útil obtenida a partir de dichos datos.

La parte del producto que es visualizada por los usuarios es una aplicación web en la cual, a partir de un sistema de roles, cada grupo de usuarios podrá tener acceso a diferentes “paneles de control” o *dashboards* relativos a todo tipo de aspectos de la ciudad. Estos paneles están enfocados principalmente a su uso por todo tipo de funcionarios, desde personal administrativo, hasta cuerpos como el de policía o el de bomberos, aunque también se dispone de algunos módulos de accesibilidad pública a la que tiene acceso cualquier ciudadano.

Durante este proyecto nos vamos a centrar en el desarrollo de un nuevo módulo que permita registrar la situación de diferentes elementos de interés en un instante concreto de tiempo para ayudar en la toma de decisiones y en el posterior análisis de la validez de ésta. Gracias al conocimiento de la situación de todo tipo de aspectos de la ciudad, muchos de ellos reflejados en tiempo real, se facilita la monitorización de situaciones de riesgo, como podría ser un incendio o un atentado terrorista, para tomar las mejores decisiones de actuación en un tiempo reducido. Este módulo está enfocado para ser utilizado principalmente por cuerpos como el de policía, bomberos o protección civil.

Anteriormente, no era posible llevar a cabo un registro tan detallado de la situación de los elementos involucrados en un acontecimiento, debido a que ni las ciudades estaban tan informatizadas, como lo están hoy en día, ni las tecnologías de las que se disponían permitían recoger tales cantidades de datos y procesarlos, prácticamente, en tiempo real.

1.3. Estructuración de la memoria

En el siguiente documento se van a describir todos los procesos llevados a cabo en cada una de las fases propias de la ingeniería de software, a excepción de la última etapa correspondiente a la documentación.

En primer lugar, se llevará a cabo una descripción en detalle de la metodología de desarrollo aplicada, así como de las diferentes tecnologías utilizadas. Seguidamente, se presentará el problema y los requisitos que deben satisfacerse para considerar el proyecto como finalizado; en tercer lugar, se detallará la arquitectura que conforma la aplicación; a continuación, se desgranará el proceso de la implementación de la solución; y finalmente, se expondrán las distintas pruebas que se han llevado a cabo para verificar el correcto funcionamiento del sistema tras implementar el nuevo módulo.

Para terminar, se incluyen una serie de conclusiones y trabajos futuros que se plantean tras terminar el desarrollo del proyecto.

2. Metodología y herramientas

En este segundo apartado se llevará a cabo una descripción de la metodología utilizada para el desarrollo del proyecto junto a las diferentes tecnologías y herramientas utilizadas para cumplir los objetivos marcados. Finalmente, se incluye el detalle de la planificación realizada.

2.1. Metodología de desarrollo

El proyecto se ha llevado a cabo siguiendo una metodología de desarrollo Iterativa e Incremental [3]. Esto quiere decir, que la planificación del proyecto ha sido dividida en una serie de iteraciones con la finalidad de tener la capacidad de mostrar al cliente funcionalidades finalizadas en diferentes momentos del proyecto, pudiendo así recibir un *feedback* más inmediato por su parte.

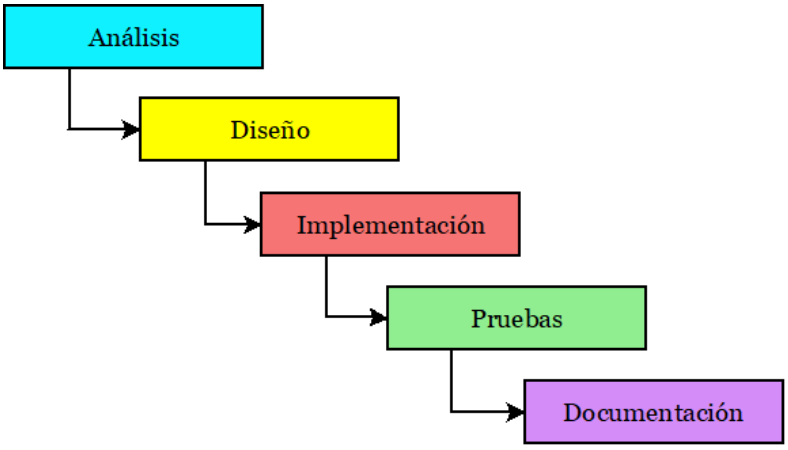

Ilustración 1. Etapas de cada iteración.

En cada una de las iteraciones, como puede observarse en la figura superior (ver Ilustración 1), se lleva a cabo cada una de las etapas propias de un desarrollo software[3]: desde el análisis previo de los requisitos y el diseño, hasta la realización final de las pruebas y documentación, pasando por la etapa de mayor peso que es la de la implementación.

En este proyecto se ha llevado a cabo una división en 3 iteraciones: una primera iteración en la cual se realiza el desarrollo de la parte de obtención de los datos de la base de datos; una segunda iteración enfocada a elaborar las

diferentes vistas que se ofrecerán al usuario; y una tercera iteración durante la cual se incluirá la configuración de los filtros predefinidos y del usuario.

Durante el desarrollo del proyecto se han llevado a cabo reuniones de forma regular con el cliente, habitualmente dos por semana, para permitirle tener un mayor seguimiento de los diferentes avances en el proyecto y poder plantear modificaciones sobre los requisitos y sugerencias de mejora a medida que se implementaban nuevas funcionalidades.

2.2. Tecnologías y herramientas utilizadas

A continuación, se describen las diferentes tecnologías que han sido utilizadas, así como las diferentes herramientas que han facilitado el uso de éstas.

2.2.1. Tecnologías

Las tecnologías utilizadas vienen dadas por el contexto en el que se lleva a cabo el proyecto, ya que es un sistema que ya se encuentra en fase de desarrollo. En su día se llevó a cabo un estudio de las diferentes tecnologías que podrían ser utilizadas para llevar a cabo la implantación de este sistema, teniendo en cuenta la velocidad de procesamiento y la escalabilidad de cada una de ellas, ya que estos aspectos son fundamentales para el correcto funcionamiento de éste. Además, también se buscaron tecnologías que supusiesen coste cero para el desarrollo y posterior mantenimiento del sistema. Es por ello por lo que se ha optado por utilizar JavaScript tanto para la parte cliente como para la parte del servidor de la aplicación web.

JavaScript

JavaScript[4] es un lenguaje de programación interpretado, lo que quiere decir que no es compilado previamente y, por tanto,

va interpretándose en tiempo de ejecución, actualmente todos los navegadores modernos poseen intérpretes de JavaScript. Inicialmente se utilizaba únicamente como un lenguaje de script en la parte web de las aplicaciones, aunque, como se verá más adelante en la memoria, en la actualidad se ha aprovechado la ligereza del lenguaje para utilizarlo también en la parte servidora.

AngularJS

AngularJS[5] es un *framework* de JavaScript enfocado a realizar aplicaciones web mucho más dinámicas y a la vez compatibles tanto con navegadores de escritorio como de dispositivos móviles. Este *framework* está destinado completamente a ser utilizado en la parte cliente de la aplicación, por lo que nos proporciona diversas opciones de sintaxis a incluir en el HTML, que, entre otras cosas, nos permiten modificar el contenido de la página web dinámicamente.

Node.js

Node.js[6] es un entorno de ejecución para JavaScript, elaborado con el motor V8 JavaScript de Google, y cuya finalidad es trasladar la programación orientada por eventos, propia de JavaScript, a la parte del servidor de las aplicaciones web. La escalabilidad y la capacidad de concurrencia aumentan notablemente gracias a que las conexiones únicamente consumen memoria del servidor cuando lanzan algún evento en éste.

HTML5

HTML[7] es un lenguaje de marcas que se apoya en etiquetas para permitir elaborar la estructura de las páginas web. HTML5[8] es la última versión de este lenguaje de marcas e incluye nuevas etiquetas y mejoras de rendimiento respecto a versiones anteriores.

CSS3

La tecnología CSS[9] es la encargada de definir los diferentes estilos de los componentes que forman la página web proporcionando una gran cantidad de opciones de customización para poder personalizar hasta el más mínimo detalle. CSS3 es la última versión de esta tecnología que incluye novedades como transiciones, gradientes o animaciones entre otras.

MongoDB

MongoDB[10] es una base de datos NoSQL de código abierto y de tipo documental. Al igual que el resto de las bases de datos NoSQL su enfoque principal es hacia la escalabilidad y el rendimiento, lo que la hacen una perfecta candidata para soluciones software en las que está presente el *big data*.

CrateDB

CrateDB[11] es una base de datos SQL distribuida de código abierto construida sobre una base NoSQL de tipo documental. Ofrece al desarrollador la capacidad de trabajar con ella como si se tratase de una base de datos SQL, pero manteniendo la escalabilidad y el rendimiento de una base de datos NoSQL.5

Git

Git[12] es un sistema de control de versiones centralizado de código abierto utilizado ampliamente en la actualidad en todo tipo de proyectos software. Como el resto de los sistemas de control de versiones, Git permite llevar un registro de todos los cambios hechos en cada uno de los ficheros del proyecto y acceder en cualquier momento a cualquiera de sus versiones. El hecho de que sea centralizado permite a los desarrolladores disponer de

un repositorio local en el cual pueden ir llevando un control de sus cambios sin necesidad de sincronizarlos con el repositorio remoto.

JSON

JSON[13] es un formato ligero utilizado para el intercambio de datos, tanto entre el cliente y el servidor, como entre el servidor y ciertas bases de datos. Su simpleza facilita tanto el trabajo de los desarrolladores como el procesado de la información.

Orion Context Broker

Orion Context Broker[27] es un importante componente que permite gestionar el ciclo de vida completo de los datos a través de una API REST proporcionando una capa de abstracción superior para la base de datos. Se encuentra dentro de la plataforma FIWARE de Telefónica enfocada al desarrollo de ciudades inteligentes.

Express.js

Express.js[28] es un *framework* de aplicaciones web para Node.js que proporciona multitud de métodos de utilidad para HTTP y *middleware* que facilitan el desarrollo aprovechando al máximo las características de rendimiento ofrecidas por Node.js.

Mocha

Mocha[31] es un *framework* pruebas para JavaScript que puede correr tanto en un entorno de Node.js como en un navegador y que facilita la ejecución de pruebas asíncronas en este tipo de entornos.

2.2.2. Herramientas

Las herramientas utilizadas para llevar a cabo el proyecto han sido aquellas que ofrecen un mayor número de facilidades para trabajar con las tecnologías seleccionadas y por tanto permiten agilizar las acciones a llevar a cabo. También se ha tenido en cuenta la experiencia previa de los integrantes del equipo con alguna de las tecnologías ya que esto reduce la curva de aprendizaje significativamente.

Visual Studio Code

Visual Studio Code[14] es un editor de código fuente de código abierto desarrollado por Microsoft y compatible con Windows, Linux y macOS. Ofrece herramientas para facilitar el desarrollo con tecnologías propias de aplicaciones web como Node.js, HTML o CSS entre otras. Este editor posee una comunidad bastante activa de desarrolladores de extensiones lo que permite disponer de un gran abanico de posibilidades para personalizar y potenciar la herramienta.

SourceTree

SourceTree[15] se trata de un entorno gráfico para trabajar con repositorios Git sin necesidad de utilizar la consola de comandos. Esta herramienta desarrollada por Atlassian es gratuita y está disponible para Windows y macOS. La interfaz gráfica ofrece grandes facilidades en el día a día, tanto para trabajar directamente con el repositorio, como para explorar el histórico de cambios en el propio repositorio.

Docker

Docker[16] es una herramienta que se encarga de llevar a cabo el empaquetamiento de las aplicaciones junto a todo el software del que son dependientes en lo que se denominan contenedores para posteriormente poder ejecutarlos en cualquier máquina que disponga de Docker.

Apache NiFi

Apache NiFi[17] es una herramienta ETL de código abierto desarrollada por Apache y dirigida a automatizar el traspaso de datos entre sistemas heterogéneos. Es decir, al igual que otras ETLs, su finalidad es la de tomar datos de fuentes de todo tipo, procesarlos para darles un formato uniforme y enviarlos hacia otro sistema.

Postman

Postman[18] se trata una herramienta enfocada al desarrollo de las API disponible para los sistemas operativos Windows, Linux y macOS e incluso dispone de una versión para el navegador Google Chrome. Postman nos permite realizar todo tipo de peticiones HTTP e incluso crear una colección de peticiones para reutilizarlas posteriormente sin necesidad de introducir todos los parámetros propios de la petición a realizar. Además, una función muy interesante de esta herramienta es la posibilidad de realizar test para comprobar el correcto funcionamiento de las API.

PuTTY

PuTTY[19] es un cliente de SSH y telnet de código abierto desarrollado originalmente para Windows pero que actualmente ha sido migrado por distintos grupos de usuarios a otros sistemas operativos como Linux o macOS.

WinSCP

WinSCP[20] es un cliente SFTP de software libre que permite llevar a cabo una transferencia segura de ficheros entre un equipo local y otro remoto que permita trabajar con servicios SSH.

Rancher

Rancher[21] es una herramienta de código abierto que proporciona una interfaz gráfica que permite llevar a cabo el despliegue y la gestión de múltiples contenedores de Docker.

Slack

Slack[22] es una herramienta de comunicación que ofrece la posibilidad de crear espacios de trabajo privados de forma gratuita.

Microsoft Project

Microsoft Project[23] es un software desarrollado por Microsoft enfocado a la administración de todo tipo de proyectos.

2.3. Planificación

El desarrollo del proyecto se ha llevado a cabo dentro de un período de prácticas externas extracurriculares realizadas en la empresa CIC Consulting Informático, comprendidas entre el 11 de febrero de 2019 y el 10 de mayo de ese mismo año.

En el diagrama de Gantt (ver Ilustración 2) incluido a continuación se puede ver reflejado el resultado de la planificación del proyecto en base a la metodología descrita en el primer apartado de este capítulo. El desarrollo de las iteraciones se precede de una toma de contacto con el proyecto, que incluye una formación inicial en las diferentes tecnologías y herramientas a ser utilizadas, así como un análisis en profundidad de los requisitos que deben satisfacerse en el desarrollo de este módulo de la plataforma.

Task Name	Duration	Start	Finish
Módulo de Enriquecimiento	**64 days**	**Mon 11/02/19**	**Thu 09/05/19**
Formación inicial	4 days	Mon 11/02/19	Thu 14/02/19
Análisis de requisitos	3 days	Fri 15/02/19	Tue 19/02/19
Iteración 1: Obtención de datos	**21,5 days**	**Wed 20/02/19**	**Thu 21/03/19**
Revisión de requisitos	0,5 days	Wed 20/02/19	Wed 20/02/19
Análisis del funcionamiento de CrateDB	3 days	Wed 20/02/19	Mon 25/02/19
Diseño detallado	3 days	Mon 25/02/19	Thu 28/02/19
Elaboración de consultas de prueba	2 days	Thu 28/02/19	Mon 04/03/19
Desarrollo del servicio generador de consultas	10 days	Mon 04/03/19	Mon 18/03/19
Pruebas	2 days	Mon 18/03/19	Wed 20/03/19
Documentación	1 day	Wed 20/03/19	Thu 21/03/19
Iteración 2: Vista del dashboard	**30 days**	**Thu 21/03/19**	**Thu 02/05/19**
Revisión de requisitos	0,5 days	Thu 21/03/19	Thu 21/03/19
Diseño detallado	1,5 days	Fri 22/03/19	Mon 25/03/19
Vista pestaña "Buscar"	**7 days**	**Mon 25/03/19**	**Wed 03/04/19**
Maquetación de la vista	2 days	Mon 25/03/19	Wed 27/03/19
Desarrollo de las funcionalidades de búsqueda	5 days	Wed 27/03/19	Wed 03/04/19
Vista pestaña "Base y filtros"	**8 days**	**Wed 03/04/19**	**Mon 15/04/19**
Maquetación de la vista	3 days	Wed 03/04/19	Mon 08/04/19
Desarrollo de la funcionalidade de consulta	2 days	Mon 08/04/19	Wed 10/04/19
Desarrollo de la funcionalidad de asociación	3 days	Wed 10/04/19	Mon 15/04/19
Vista pestaña "Enriquecimientos asociados"	**4 days**	**Mon 15/04/19**	**Fri 19/04/19**
Maquetación de la vista	1 day	Mon 15/04/19	Tue 16/04/19
Carga de datos en la tabla	1 day	Tue 16/04/19	Wed 17/04/19
Desarrollo de la funcionalidad de visualización y desasociación	2 days	Wed 17/04/19	Fri 19/04/19
Vista pestaña "Enriquecimientos no asociados"	**2 days**	**Fri 19/04/19**	**Tue 23/04/19**
Maquetación de la vista	1 day	Fri 19/04/19	Mon 22/04/19
Carga de datos en la tabla	1 day	Mon 22/04/19	Tue 23/04/19
Pruebas	5 days	Tue 23/04/19	Tue 30/04/19
Documentación	2 days	Tue 30/04/19	Thu 02/05/19
Iteración 3: Configuración de filtros predeterminados	**5,5 days**	**Thu 02/05/19**	**Thu 09/05/19**
Revisión de requisitos	0,5 days	Thu 02/05/19	Thu 02/05/19
Diseño detallado	1 day	Thu 02/05/19	Fri 03/05/19
Añadir filtros predeterminados por el tipo de incidencia	2 days	Fri 03/05/19	Mon 06/05/19
Añadir filtros guardados por el usuario	1 day	Tue 07/05/19	Tue 07/05/19
Pruebas	1 day	Wed 08/05/19	Wed 08/05/19
Documentación	1 day	Thu 09/05/19	Thu 09/05/19

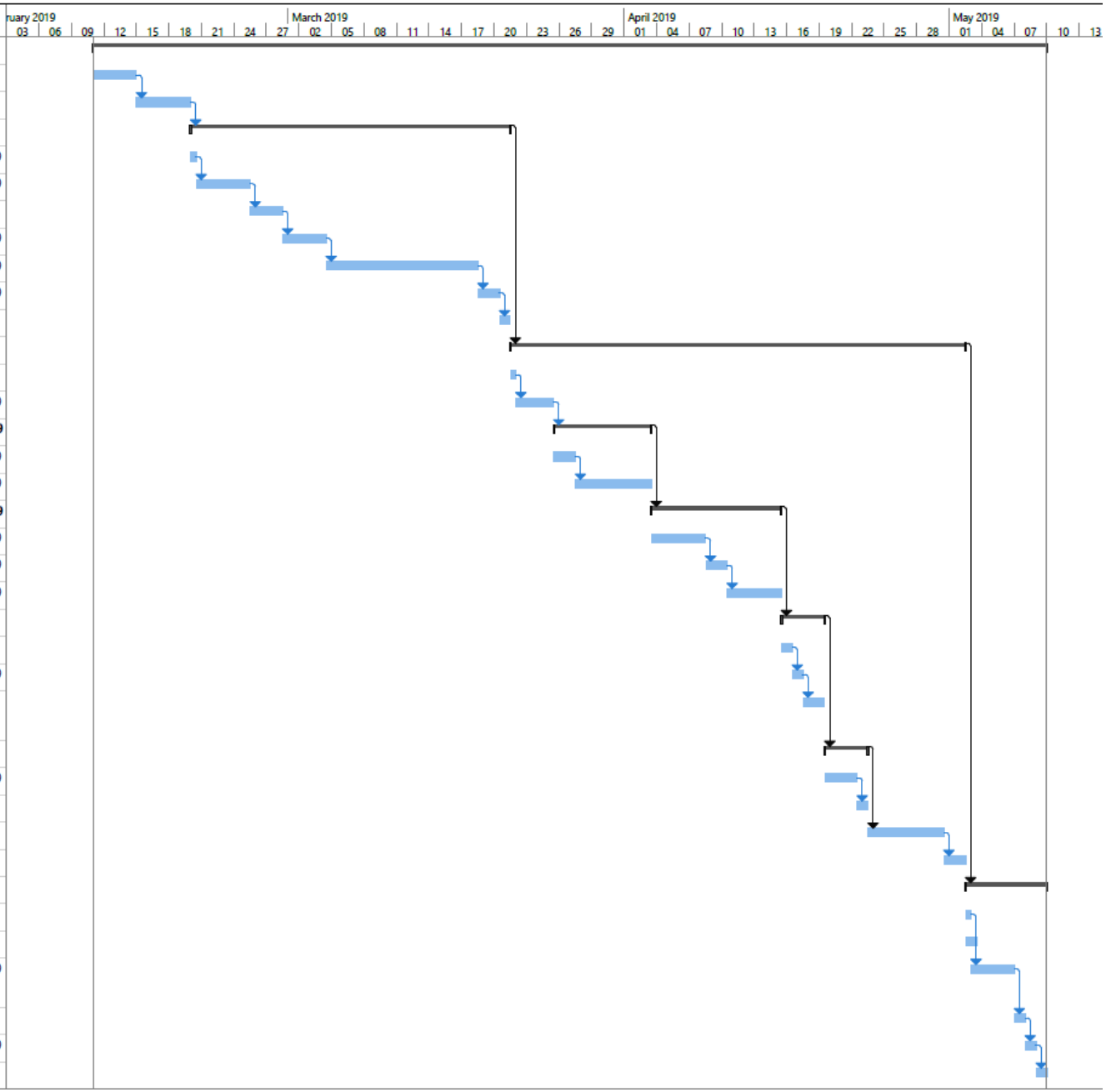

Ilustración 2. Diagrama de Gantt.

3. Presentación del problema y análisis de requisitos

Este tercer apartado de la memoria recoge una descripción de las diferentes características que ofrecerá la aplicación al usuario tras el desarrollo del proyecto, así como los requisitos funcionales y no funcionales definidos por este.

3.1. Planteamiento del problema

Durante la introducción se ha indicado que la finalidad de la realización de este proyecto es la de aprovechar al máximo la información de la que se dispone sobre los elementos que rodean un suceso para poder estudiar y mejorar la toma de decisiones.

A continuación, se listan las diferentes vistas de las que dispondrán los usuarios para poder alcanzar dicha meta. En las ilustraciones correspondientes a las vistas se han camuflado tanto los datos como los mapas por motivos de privacidad sobre el desarrollo del proyecto por parte de la empresa.

3.1.1. Vista de búsqueda de elemento base

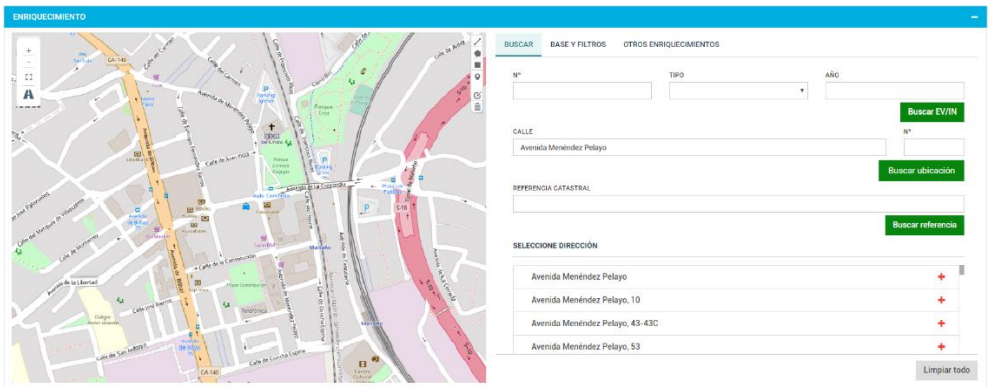

Ilustración 3. Vista de búsqueda de elemento base.

En primera instancia, el usuario dispondrá de una vista (ver Ilustración 3) en la que llevará a cabo la selección del elemento que desea utilizar como referencia para ser enriquecido. La búsqueda de este elemento base puede llevarse a cabo tanto a través del identificador de un evento o incidente ya registrado en la plataforma

previamente, como de una referencia geográfica dentro de la ciudad, esta segunda puede ser obtenida de dos formas diferentes: a través de la búsqueda de la calle y/o el número del edificio y a través de la referencia catastral. El usuario también podrá apoyarse en la búsqueda de direcciones a través de la selección de una zona en el mapa.

3.1.2. Vista combinada, elemento base y filtros

Una vez se dispone de un elemento base sobre el cual se quiere obtener información acerca de los elementos que lo rodean, se accede a esta segunda vista del módulo. Esta vista permite al usuario obtener la información que desea a través de la combinación de diversos filtros donde se le permite indicar: la distancia en metros desde la geometría asociada al elemento base, un espacio temporal (el momento actual en caso de no indicarlo) y la selección de los tipos de elementos a visualizar.

En las siguientes dos figuras (ver Ilustraciones 4 y 5) podemos observar la distribución de los diferentes elementos de filtrado precedida por la información de la entidad que se está utilizando como referencia.

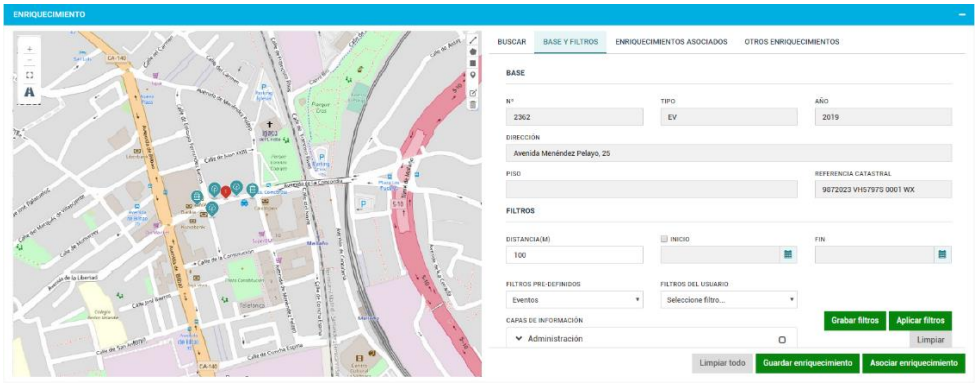

Ilustración 4. Vista combinada, elemento base filtros (1).

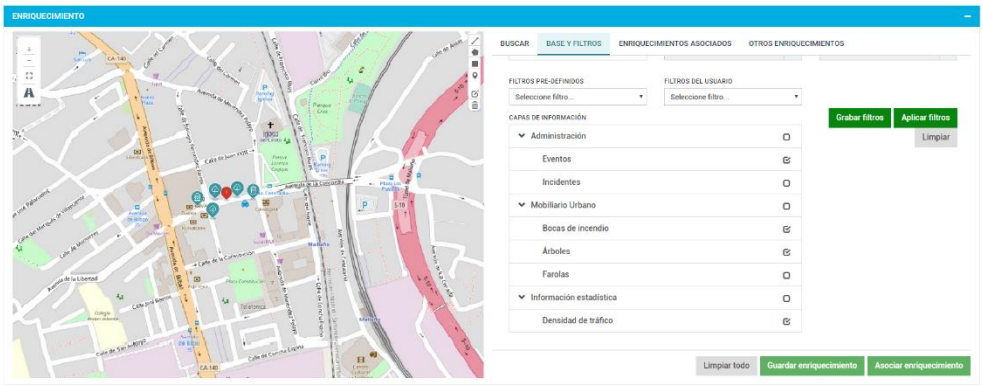

Ilustración 5. Vista combinada, elemento base filtros (2).

También se ofrece la posibilidad de trabajar con combinaciones de filtrado ya guardadas previamente; en la plataforma, a través de filtros predefinidos para ese tipo de elemento; o por el usuario, pudiendo utilizar filtros previamente guardados por el mismo.

A partir de ahora, el conjunto de datos obtenido de la aplicación de los filtros se referenciará como enriquecimiento, ya que éste es el término utilizado en la plataforma para hacer referencia a esta captura de información. En esta vista se podrá guardar dicho enriquecimiento para su posterior asociación a una entidad (evento o incidente) o asociarlo directamente a la entidad utilizada como base, siempre y cuando no se trate de una referencia geográfica, en cuyo caso se dispondrá únicamente de la opción de guardado para su posterior asociación.

3.1.3. Vista de enriquecimientos asociados

Esta tercera vista (ver Ilustración 6) permite visualizar de forma individualizada o conjunta los diferentes resultados de búsquedas que se encuentran asociados al elemento que se está utilizando como referencia. Si se llevase la simplificación al máximo, podría considerarse esta vista como el conjunto de capturas de instantes concretos de tiempo de los elementos de interés que rodean al elemento, lo que puede resultar de gran utilidad para recomponer la evolución de los hechos y estudiar si se tomaron las decisiones

adecuadas. También nos permite desasociar búsquedas que se asociaron al elemento que ya no se consideran de relevancia.

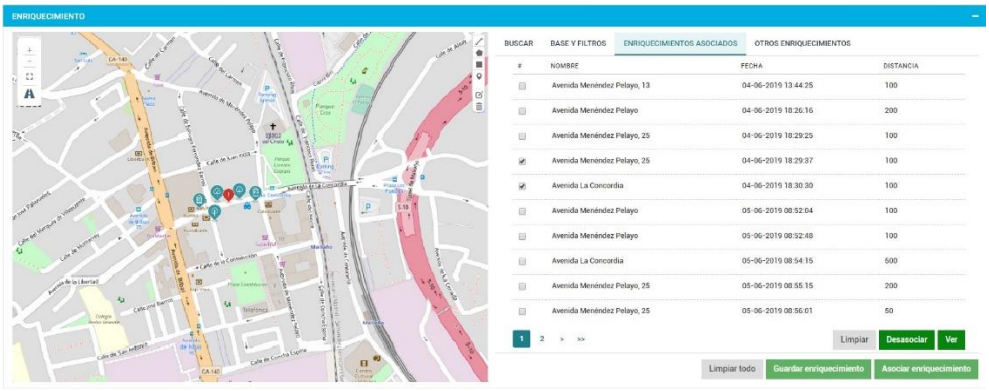

Ilustración 6. Vista de enriquecimientos asociados.

3.1.4. Vista de otros enriquecimientos

Por último, se dispone de una vista (ver Ilustración 7) que recoge el conjunto de enriquecimientos que han sido desasociados por el usuario o que han sido guardados sobre elementos geográficos y no sobre entidades propiamente dichas, como son los eventos y las incidencias.

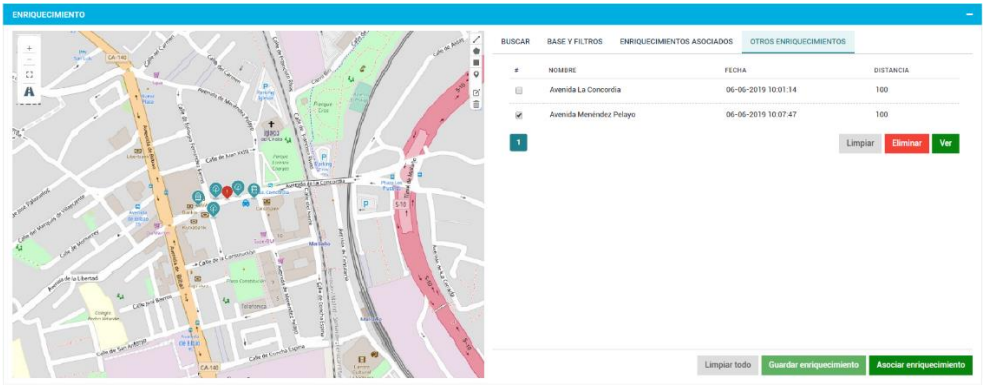

Ilustración 7. Vista de otros enriquecimientos.

Desde esta vista se pueden visualizar o eliminar dichos enriquecimientos, así como asociarlos a la entidad que estamos utilizando como base, siempre y cuando no se trate de una referencia geográfica.

3.2. Requisitos funcionales

Una vez se han descrito las diferentes vistas que compondrán el módulo, se pasa a enumerar los diferentes requisitos que han sido acordados con el cliente para su desarrollo.

Los requisitos funcionales[24] recogen las funcionalidades que el sistema debe proporcionar a los usuarios, su respuesta frente a situaciones específicas y, a veces, incluso indican cómo no debe comportarse éste.

En las siguientes tablas aparecen detallados los diferentes requisitos del sistema que tienen una implicación directa con el módulo a desarrollar:

Código	RF/001.
Módulo	Enriquecimiento.
Nombre	Definición.
Requisito	El enriquecimiento de la información es un proceso de búsqueda y cruce de información basado en criterios de espacio y tiempo diseñado para facilitar la toma de decisiones.

Código	RF/002.
Módulo	Enriquecimiento.
Nombre	Categorización.
Requisito	Deberán existir dos tipos de enriquecimiento: • Automático: es generado a partir de los filtros predefinidos para la tipología asociada al evento o incidente sobre el que se esté trabajando. El operador decidirá si desea mantener o desechar estos datos. • Manual: realizado por el operador seleccionando manualmente los filtros que desea incluir, bien desde cero o bien partiendo de alguno de los filtros predefinidos.

Código	RF/003.
Módulo	Enriquecimiento.
Nombre	Funcionalidades básicas.
Requisito	La plataforma deberá permitir las siguientes acciones: • La fácil configuración y asociación de conjuntos de filtros predefinidos a las diferentes tipologías de eventos e incidentes.

	• El acceso a las opciones de filtrado por el operador permitiéndole llevar a cabo el proceso de cruce de información desde cero o partiendo de filtros predefinidos. • El guardado de filtros definidos por el propio usuario para su posterior reutilización. Estos filtros sólo deben ser visibles por el usuario que los crea. • Llevar a cabo el enriquecimiento automático de información en los eventos e incidentes en base a los filtros predefinidos para el tipo que tengan asociado. Actualizándose en caso de que se modifiquen datos como la fecha o la ubicación del suceso. • Permitir la asociación y desasociación del resultado de los enriquecimientos, tanto manuales como automáticos, de los diferentes eventos e incidentes. • Permitir el fácil acceso por parte del operador a los diferentes enriquecimientos realizados.

Código	RF/004.
Módulo	Enriquecimiento.
Nombre	Campos.

Requisito	Los campos base que deberá incluir el módulo de enriquecimiento son los siguientes:
	• Localización: definido por la calle, la referencia catastral o la señalización directa sobre el mapa de un punto, línea o área.
	• Filtros:
	  ○ Definición de un radio espacial alrededor de la localización seleccionada.
	  ○ Definición de un marco temporal (día y hora) a través de la introducción de las fechas de inicio y fin.
	  ○ Conjunto de capas que podrán ser seleccionadas para la visualización por parte del operador.
	• Las herramientas restantes para facilitar el desarrollo de la tarea, las cuales incluyen:
	  ○ Limpiar: permite anular la selección de filtros activos.
	  ○ Optimización de rutas: permite obtener rutas alternativas entre dos o más localizaciones y sus respectivos tiempos de desplazamiento.

3.3. Requisitos no funcionales

Tras haber descrito los requisitos funcionales del sistema se procede a enumerar los diferentes requisitos no funcionales que lo componen, ya que ambos influyen en las posteriores fases del desarrollo.

Los requisitos no funcionales[24], a diferencia de los funcionales, definen características que no tienen que ver directamente con el uso que van a hacer los usuarios del sistema, si no que son características que debe cumplir el sistema en conjunto para permitir a los usuarios aprovecharse de las funcionalidades que se les ofrece.

Estos requisitos no funcionales se suelen clasificar[24] en tres grandes grupos: del producto, especifican propiedades mínimas y limitadoras que debe poseer el sistema; de la organización, relativos a los procesos y herramientas que se deben seguir según las políticas de la organización del cliente; externos, principalmente incluyen valores éticos y requisitos legales que debe cumplir el producto.

En el diagrama que se incluye a continuación (ver Ilustración 8), podemos observar un desglose algo más amplio de estos tres grupos para comprender qué tipo de ámbitos abarca cada uno de ellos.

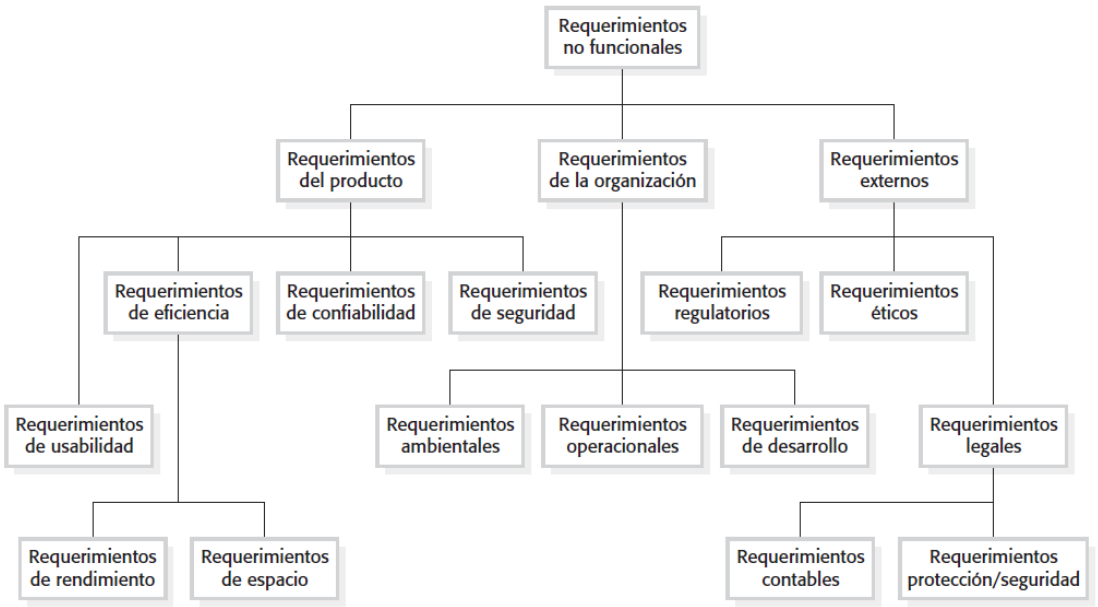

Ilustración 8. Clasificación de requisitos no funcionales.

A continuación, se indican algunos de los requisitos no funcionales del grupo de producto que se han establecido para el desarrollo de esta plataforma.

Código	RNF/001.
Nombre	Comunicaciones / Acceso.
Requisito	Las interfaces de usuario disponibles se producirán sobre tecnologías Web, soportando HTTP y su versión segura HTTPS.

Código	RNF/002.
Nombre	Disponibilidad.
Requisito	La disponibilidad se calcula al final de cada uno de los semestres (entre el 2º y el 6º semestre) mediante la aplicación de la siguiente fórmula: **D[%] = [(P – H + HIP) / P] x 100** **P** – Número de horas de funcionamiento potencial en el período considerado; **H** – Número de horas de indisponibilidad en el período considerado; **HIP** – Número de horas de indisponibilidad programada durante el período considerado; La indisponibilidad programada se deriva de la necesidad de ejecución de servicios de mantenimiento, reconfiguración o actualización de los componentes, siempre que se hayan comprobado las condiciones siguientes: 1. La interrupción del funcionamiento no exceda de 30 minutos semanales, o de forma acumulada 2 horas anuales. 2. La interrupción programada de funcionamiento, de conformidad con el apartado anterior, depende de una previa comunicación escrita, dirigida por el proveedor al organismo con una antelación mínima de 5 días en relación con la fecha prevista para la interrupción. 3. En caso de indisponibilidad no imputable al adjudicatario, éste se obliga a comunicar por escrito, y demostrar documentalmente, el motivo de estos eventos ante el organismo. 4. Podrán existir otras situaciones excepcionales de intervenciones programadas que, ante el acuerdo previo

	con el organismo, excedan los tiempos especificados anteriormente. 5. La disponibilidad, calculada para períodos de 6 meses, tendrá que ser superior al 99,5%.

Código	RNF/003.
Nombre	Accesibilidad.
Requisito	La interfaz Web deberá ser accesible siguiendo las pautas marcadas por las *Web Content Accesibility Guidelines (WCAG) 2.0 / Compliance Level AA*[29].

Código	RNF/004.
Nombre	Copias de seguridad.
Requisito	Se tendrá que aplicar una estrategia de tolerancia a fallos, basada en las siguientes premisas: 1. Tendrá que ser posible realizar copias de seguridad integrales y parciales sin interrumpir el funcionamiento. 2. Deberá ser posible restablecer las respectivas copias de seguridad, así como hacer disponible el sistema en producción a partir de las mismas copias en un tiempo máximo de 1 hora. 3. Todos los procedimientos tendrán que ser efectuados por el equipo de la organización de acuerdo con la documentación proporcionada. 4. Los procedimientos de gestión de fallos deben formalizarse, en particular, implementando los procedimientos de simulación que confirmen la eficiencia en la recuperación de los servicios.

Código	RNF/005.
Nombre	Componentes de terceros.
Requisito	Cuando se requieran componentes *software* de terceros, estos nunca podrán requerir licencias adicionales. Algunos ejemplos de componentes *software* de terceros son el servidor Web, el servidor de aplicaciones o el sistema gestor de base de datos.

Código	RNF/006.
Nombre	Capacidad de la infraestructura computacional.
Requisito	La plataforma suministrada se ejecutará en la infraestructura computacional del organismo con los siguientes recursos: • 2x Hosts con 64GB de RAM cada uno, con CPU de 28 *cores*. • 50TB de almacenamiento.

Código	RNF/007.
Nombre	Límites/Licencias.
Requisito	La plataforma suministrada no podrá estar limitada por licenciamientos a través de ningún mecanismo de control de carga o del número de accesos a las API.

Código	RNF/008.
Nombre	Sistema para el registro de incidencias.
Requisito	Se deberá proporcionar una plataforma para el registro de incidentes y planificación de los períodos de mantenimientos correctivos.

Código	RNF/009.
Nombre	Prioridades de los pedidos de soporte.
Requisito	Se deberán de actuar ante los pedidos de soporte siguiendo las siguientes prioridades: 1. **Prioridad 1**: *Tiempo de inactividad crítico* – Implica la indisponibilidad de la plataforma, riesgo de pérdida de datos o inconsistencia de estos, hace que sea imposible para la organización llevar a cabo su misión. 2. **Prioridad 2**: *Degradación del servicio* – El entorno operativo se ve afectado, existe un alto riesgo de indisponibilidad, impacto significativo en la misión. 3. **Prioridad 3**: *Normal* – No tiene impacto en el funcionamiento de la plataforma.

Código	RNF/010.
Nombre	Tiempo de respuesta a los pedidos de soporte.
Requisito	Se deberán de actuar ante los pedidos de soporte dentro de los siguientes tiempos:

	Tiempo máximo de confirmación de recepción del incidente e inicio de la corrección	**Tiempo máximo de resolución**
Prioridad 1	1 hora	4 horas
Prioridad 2	1 hora	8 horas
Prioridad 3	2 horas	48 horas

3.4. Permisos y roles

El uso de este módulo de la aplicación está limitado a algunos usuarios de la plataforma y no todos tienen los permisos necesarios para realizar las mismas acciones.

Para poder entender cómo se limitan dichos permisos en primero lugar hay que entender cómo funcionan los permisos a nivel de la aplicación. Los usuarios dentro de la plataforma están categorizados en diferentes grupos denominados roles, existiendo la posibilidad de que un único usuario tenga asignados varios roles diferentes. A su vez, los roles tienen asignados una serie de permisos que son los que les proporcionan la capacidad de poder realizar acciones concretas dentro de la plataforma. Por ejemplo, puede existir un rol denominado *"operario_protección_civil"* que entre sus permisos tenga uno denominado *"asociar_enriquecimientos"* que le permite llevar a cabo la asociación de enriquecimientos a eventos e incidentes dentro de la plataforma.

Por otra parte, las capas visibles para seleccionar en las opciones de filtrado pueden verse limitadas por los propios roles, un ejemplo de ello es la capa de visualización del posicionamiento de las patrullas de policía cuya información sólo estará disponible para aquellos roles que tengan una relación directa con la policía y necesiten hacer uso de esta información.

4. Diseño

Después de haber profundizado algo más en el problema y de haber detallado los diferentes requisitos, tanto funcionales como no funcionales definidos para el sistema y, en concreto, para el módulo a desarrollar, se procede a detallar el diseño utilizado para alcanzar la solución.

4.1. Arquitectura de la aplicación

Durante el diseño de la plataforma, como es habitual cuando se utiliza el *framework* AngularJS, se ha tomado la decisión de seguir el patrón de diseño MVC[25]. Dicho patrón define una serie de pautas para la estructuración del proyecto que separan la lógica de negocio de la lógica de la vista facilitando así la organización y por tanto la mantenibilidad de éste.

El patrón MVC define tres componentes diferentes y las funcionalidades de cada uno de estos:

- **Modelo**. El modelo es la capa más profunda de la aplicación y desconoce la existencia de las capas superiores. Esta capa es la encargada de gestionar los datos comunicándose con el servidor de bases de datos.
- **Vista**. La vista recoge todos los aspectos relativos a la presentación que se hace al usuario de los datos.
- **Controlador**. El controlador es la parte de la aplicación donde reside la lógica de negocio y sirve de conexión entre las otras dos capas del sistema.

Siguiendo este patrón se consigue una reducción del acoplamiento en el sistema y por tanto se facilita la edición de cualquiera de los componentes sin necesidad de modificar otras capas.

4.2. Diagrama de despliegue

A continuación, se incluye el diagrama de despliegue de la aplicación (ver Ilustración 9), el cual representa la distribución de los diferentes componentes del sistema y las vías de comunicación abiertas entre ellos.

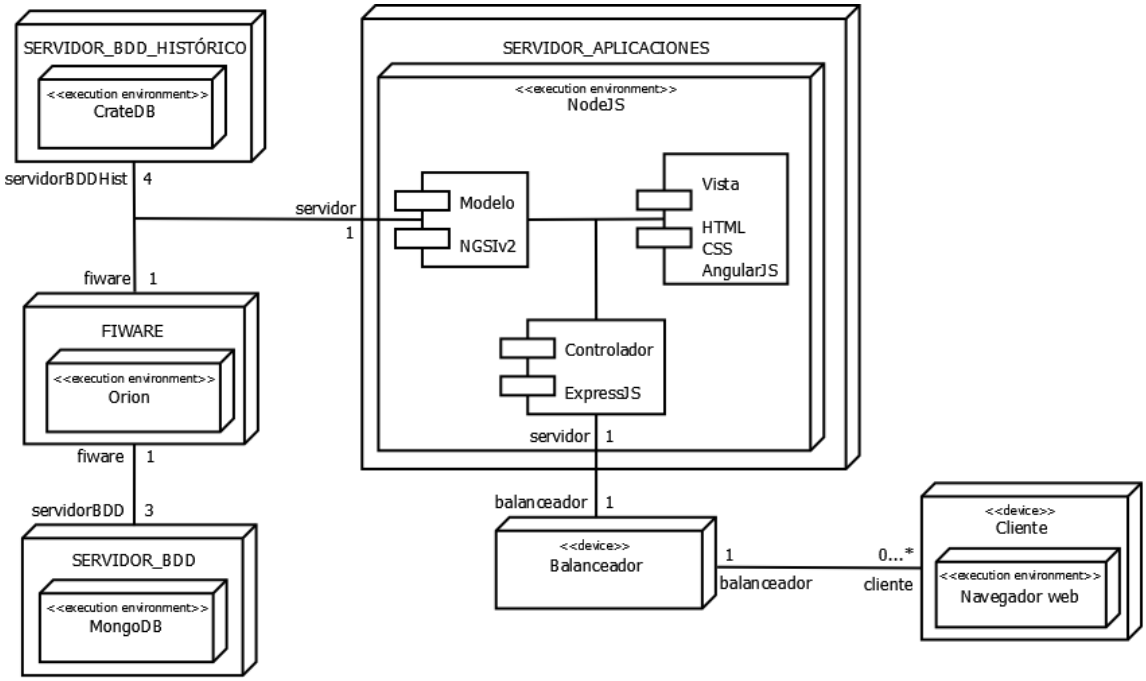

Ilustración 9. Diagrama de despliegue del sistema.

Cómo se puede apreciar en el diagrama, el núcleo de la aplicación contiene la aplicación web, con los componentes descritos en el apartado anterior, ejecutándose sobre un entorno Node.js. El *framework* utilizado para trabajar en la parte servidora de dicho entorno es Express.js, el cuál es prácticamente el estándar y nos ofrece diferentes funcionalidades que facilitan y reducen el tiempo de desarrollo.

Este servidor de aplicaciones presenta dos conexiones hacia las fuentes de datos:

1. Hacia un servidor que aloja un *middleware* de tipo Fiware denominado Orion Context Broker encargado de verificar que la estructura de los datos cumple con el estándar NGSIv2 (se detallará en el próximo capítulo), y de facilitar el trabajo con la base de datos MongoDB y de

replicar todas las operaciones de creación, modificación y borrado en la base de datos CrateDB que actúa como histórico de la aplicación.

2. Hacia el servidor de base de datos histórico para llevar a cabo operaciones de lectura de datos para así cumplir con el propósito de permitir a los usuarios conocer el estado en el que se encontraban diferentes elementos de la ciudad en un marco temporal cerrado, ya que la base de datos principal únicamente contiene el estado actual de las entidades. Cómo se puede observar ambos servidores de base de datos se encuentran alojados en varios nodos para repartir la carga de trabajo durante la gran ingesta de datos que se llevan a cabo en ciertas situaciones.

Hacia la parte del cliente, el servidor de aplicaciones utiliza como intermediario un balanceador de carga para regular las peticiones recibidas y evitar así poner en riesgo la estabilidad del sistema, permitiendo garantizar el cumplimiento de los requisitos no funcionales referidos a la disponibilidad de la aplicación.

4.3. Diagrama de componentes

Para finalizar, se incluye a continuación un diagrama de componentes (ver Ilustración 10) en el que se refleja únicamente la parte del sistema relativa a este módulo, ya que el resto de los componentes no tienen relevancia para entender su diseño.

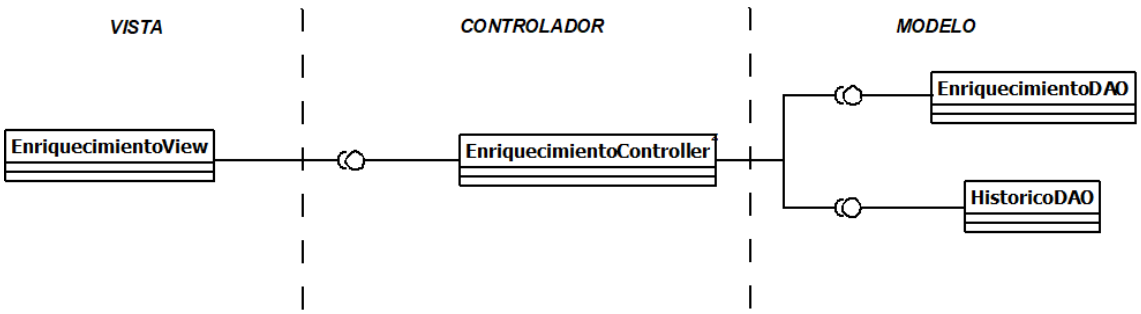

Ilustración 10. Diagrama de componentes del módulo.

Como se puede observar, la división en componentes del módulo es bastante simple y, salvo en el caso del modelo, está compuesto prácticamente por un elemento en cada una de las capas definidas en la arquitectura.

Cabe destacar que en el caso de la capa relativa al modelo sí que existe una división en dos componentes, uno referente a la persistencia de los enriquecimientos (con Orion), el otro se ocupa de trabajar con la base de datos del histórico para la recuperación de datos aplicando las opciones de filtrado seleccionadas.

5. Implementación

En este quinto capítulo se va a detallar la implementación de la solución y la implicación de cada una de las capas y los diferentes sistemas que van a trabajar de manera conjunta para poder satisfacer las necesidades del cliente.

En el anterior apartado de la memoria se detalló la arquitectura del sistema la cual ha sido implementada utilizando la tecnología Docker. Cada una de las partes del sistema que se han podido identificar en el diagrama de arquitectura de la aplicación se encuentra desplegada en un contenedor Docker permitiéndolas así trabajar de manera independiente y que un fallo en alguna de ellas no afecte al resto del sistema. Dichos contenedores se gestionan a través de una interfaz gráfica denominada Rancher a través de la cual se han configurado las diferentes conexiones entre ellos para permitir su comunicación.

5.1. Ingesta de datos

En primer lugar, se va a describir el funcionamiento de la ingesta (*intake*) de datos, crucial para entender de manera correcta el funcionamiento de este módulo.

Los datos son insertados al sistema a través de una herramienta ETL denominada Apache NiFi que se encarga de obtenerlos de numerosas fuentes y procesarlos para almacenarlos en la base de datos de la plataforma. Dentro de esta herramienta se han definido los flujos correspondientes a cada fuente de datos y se han configurado los momentos en los que estos serán consultados. Finalmente, la ETL transmite los datos al *middleware* en formato JSON cumpliendo con el modelo definido en el estándar NGSIv2 para su inserción, tanto en la base de datos principal, MongoDB, como en la base de datos relativa al histórico, CrateDB.

La principal diferencia entre ambas bases de datos es que, en el caso de la primera, si se envía una entidad con el mismo identificador que otra ya existente, ésta será actualizada en la base de datos, mientras que en el caso de la base de datos que contiene el histórico, se insertará la entidad como un nuevo registro

en su tabla correspondiente (según el tipo) y se actualizará el registro anterior correspondiente a la entidad de ese identificador, rellenando la columna correspondiente a la fecha de validez con la fecha actual, para indicar hasta que momento ese registro fue el más reciente para dicha entidad.

5.2. Modelo

La capa relativa al modelo es la encargada de llevar a cabo la comunicación con las diferentes bases de datos, de ahí a que el modelo en este módulo se haya divido en dos componentes, para recuperar la información deseada.

En la parte del modelo tiene una gran importancia cumplir con las directivas marcadas por la especificación FIWARE-NGSI v2[30] de Telefónica que indican como administrar el ciclo de vida de los datos. Cumplir con esta especificación es un requisito clave en el proyecto, ya que se plantea la idea de que en un futuro las ciudades inteligentes compartan datos entre ellas, y por tanto es importante desde el principio estandarizar la estructuración de los datos.

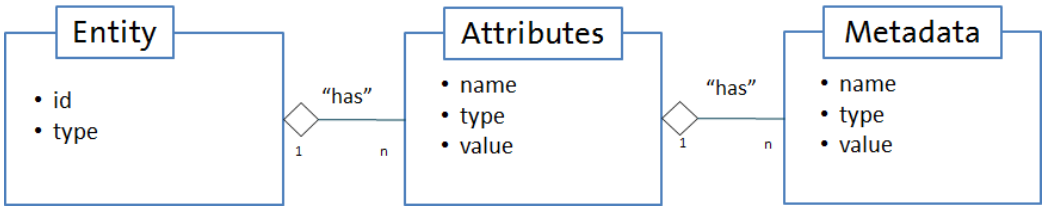

Ilustración 11. Definición de elemento según el FIWARE-NGSI v2.

La especificación mencionada (ver Ilustración 11) indica que todas las entidades deben seguir un formato JSON y contener los campos "*id*" y "*type*", actuando este segundo como si fuese el nombre de la tabla en la cual se insertaría dicha entidad. Además, marca las pautas que deben seguir los atributos de posicionamiento geográfico, así como otras peculiaridades. El encargado de verificar el cumplimiento de estas pautas es el Orion Context Broker que actúa de capa de abstracción por encima de las bases de datos. Todas las llamadas para la persistencia y actualización de entidades se harán a través de este *middleware* y el único caso en el que se preguntará directamente a la base de datos será para realizar consultas sobre la base de datos del

histórico ya que, al no realizarse ningún cambio sobre los datos, no es necesaria ninguna comprobación intermedia y por tanto el rendimiento es mucho mayor.

Por último, en la parte del modelo relativa al histórico se construyen las consultas a la base de datos en base a los filtros seleccionados por el usuario, donde la mayor complejidad reside en el filtrado espacial. Para ello, se ha utilizado la librería Turf.js que proporciona gran variedad de funcionalidades para trabajar con geometrías. Como se puede comprobar en el extracto de código incluido a continuación (ver Ilustración 12), para generar una extensión de una geometría (en metros indicados por el usuario en el filtro), basta con realizar una llamada al método *buffer()* de la librería, pasándole como parámetros la geometría inicial y los metros que la deseamos expandir. Posteriormente, esta geometría resultante será utilizada en la consulta para utilizarla como condición de intersección y poder obtener los datos que cumplen con el resto de los filtros seleccionados y o bien intersecan o bien están contenidos dentro del área delimitado.

```
function createMultipolygon(geometryCollection, bufferInMeters) {
  var geometry = buffer(geometryCollection, bufferInMeters / 1000);
  var multipolygon = {
    "type" : "MultiPolygon",
    coordinates : []
  }
  if(geometry.type == "FeatureCollection") {
    for(var i=0;i<geometry.features.length; i++) {
      var feature = geometry.features[i];
      multipolygon.coordinates.push(feature.geometry.coordinates);
    }
  } else {
   multipolygon.coordinates.push(geometry.geometry.coordinates);
  }
  // A veces el multipolygon generado contiene 2 arrays en vez de 1 lo que da un error al intentar castear.
  // El siguiente bucle corrige la estructura para evitar que de error.
  for(var i = 0; i <  multipolygon.coordinates.length ; i++)
  {
    if( multipolygon.coordinates[i].length>1)
    {
      var copiaArray =  multipolygon.coordinates[i].slice()
      multipolygon.coordinates.splice(i, 1);
      for(var e = 0; e <  copiaArray.length ; e++)
      {
        multipolygon.coordinates.push(copiaArray[e]);
      }

      i--;
    }
  }
  return multipolygon;
}
```

Ilustración 12. Método generador del área de filtrado.

5.3. Controlador

El controlador, como se ha indicado en el capítulo relativo al diseño, es el contenedor de la lógica de negocio de la aplicación. Desde el controlador se lleva a cabo la comunicación con el modelo para las diferentes operaciones de consulta y persistencia a través de peticiones AJAX a la API expuesta en el servidor de Node.js.

En primer lugar, durante la carga del módulo se comprobarán los permisos que posee el usuario para realizar diferentes acciones, como pueden ser crear enriquecimientos o asociarlos a la entidad seleccionada como base. A continuación, se solicitará el árbol de capas con las opciones correspondientes a los roles que posee el usuario, ya que algunas capas serán exclusivas de ciertos roles.

A continuación, se lleva a cabo una breve descripción de cada una de las funcionalidades que se han implementado en este módulo y cuya lógica reside en esta capa de la aplicación:

- **Selección de elemento base:** permite la búsqueda del elemento que se quiere utilizar como base del enriquecimiento a través de varias opciones:
    - **Por identificador:** introduciendo el identificador del incidente o evento que se quiere seleccionar. En función de la existencia y la visibilidad del usuario sobre el elemento indicado, se obtendrá resultado en la búsqueda o no.
    - **Por dirección:** permite obtener un listado de direcciones a partir de la introducción completa o parcial de una dirección para posteriormente seleccionar cuál de ellas se quiere utilizar como base.
    - **Por referencia catastral:** ofrece la posibilidad de introducir la referencia catastral de un inmueble o una parcela para utilizarlo como base.
    - **Por selección en el mapa:** la funcionalidad de selección en el mapa es la de llevar a cabo la búsqueda de direcciones en la zona

seleccionada para mostrar la misma lista de resultados que la obtenida utilizando la búsqueda por dirección.

- **Filtrado de elementos próximos:** se recogen los valores introducidos en los diferentes filtros ofrecidos al usuario para, utilizando dichos valores, construir la petición de búsqueda que se hace al modelo. El modelo devuelve el conjunto de elementos que se encuentran dentro de los criterios de filtrado indicados para que sean representados en el mapa. Los filtros que pueden ser rellenados son los siguientes:
    - **Distancia:** rango en metros alrededor de la entidad base que se quiere utilizar como intersección en la búsqueda.
    - **Rango temporal:** existen dos selectores de fecha con precisión de minutos para que el usuario pueda definir el marco temporal sobre el que quiere realizar la búsqueda, en caso de no rellenar este filtro se obtendrían los estados más actuales de los datos.
    - **Capas de información:** este filtro es obligatorio ya que se seleccionan los tipos de elementos que van a ser filtrados y que no tendría sentido mantener vacíos. Se ofrecen en forma de árbol los distintos conjuntos de tipos para permitir una selección ágil de grupos completos.
- **Filtros predefinidos:** cuando se lleva a cabo la selección de la entidad base, en caso de tratarse de un incidente o un evento, se llevará a cabo una petición para obtener los filtros asociados al tipo de la entidad. Por ejemplo, si se selecciona una incidencia con alguno de los tipos definidos para incendios, se cargarán en el combo filtros relativos a incendios con valores predefinidos para cada uno de los campos relativos a la distancia y la selección de capas de información.
- **Filtros del usuario:** se permite al usuario en cualquier momento dar un alias a su configuración actual de los filtros para poder acceder fácilmente a ella desde cualquier tipo de elemento base a través de su selección en un combo.
- **Limpiar filtros:** es posible vaciar los valores de los campos de filtrado a través de un botón.

- **Limpiar todo:** se limpian todas las selecciones del módulo, menos la del elemento base en el caso de haber inicializado la directiva con un elemento base ya seleccionado.
- **Visualizar enriquecimientos:** el usuario puede seleccionar tanto enriquecimientos asociados a la entidad que se está utilizando como base, como los no asociados para visualizar el conjunto de datos que contienen, así como los filtros que fueron utilizados para obtener dicho resultado.
- **Guardar enriquecimiento:** se compone la entidad relativa al enriquecimiento incluyendo en ella los identificadores de las entidades mostradas en el mapa y las opciones rellenadas en los filtros para su posterior recuperación. No se rellena el campo relativo a la asociación a una entidad base.
- **Asociar enriquecimiento:** en el caso de estar visualizando uno o varios enriquecimientos de la pestaña de enriquecimientos no asociados y encontrarse el usuario con una entidad base que sea, bien un incidente o bien un evento, podrá utilizar el botón de este mismo nombre para actualizar dicho enriquecimiento e incluirle la asociación a la entidad base.
- **Desasociar enriquecimiento:** se ofrece un botón con la funcionalidad opuesta a la anterior para el caso en el que se estén visualizando enriquecimientos ya asociados a la entidad, poder eliminarla.

5.4. Vista

En último lugar, la vista es la encargada de presentar los datos de manera visual al usuario y permitirle interaccionar con los diferentes elementos que la componen. Para el desarrollo de la plataforma se ha utilizado el *framework* AngularJS, que proporciona numerosas funcionalidades que facilitan el dinamismo de la página y el intercambio de información entre la vista y el controlador. Algunas de las funcionalidades más relevantes de AngularJS son:

- **Two Way Data-Binding:** una de las características principales de este *framework* es la notificación de cambios entre la vista y el controlador de

manera automática, lo que permite que, si se realiza cualquier modificación en alguna de las capas, la otra estará en permanente conocimiento de los cambios y podrá llevar a cabo las acciones que desee.

- **Directivas:** permite encapsular elementos HTML en directivas para, posteriormente, reutilizarlos a través de etiquetas. Esta funcionalidad permite reutilizar gran cantidad de código y facilitar la mantenibilidad de la aplicación. En la plataforma, una directiva a la se recurre constantemente es la del mapa, definiendo mediante parámetros que funcionalidades de éste se desean ofrecer en cada situación.
- **Interpolación:** ofrece la posibilidad de interpolar en el HTML, tanto expresiones algebraicas como atributos de las entidades, que además serán actualizadas en tiempo real gracias al Two Way Data-Binding mencionado en el primer punto. La interpolación se lleva a cabo encapsulando la expresión entre una doble llave. A continuación, se indican algunos ejemplos:
    - Resultado 3+3: {{3+3}} => Resultado 3+3: 6
    - Alias {{user.nickname}} => Alias dpn285
- Inyección de dependencias: es capaz de gestionar la inyección de dependencias, ya sea de servicios propios de la librería *core* del *framework,* como de servicios personalizados incorporados en la plataforma.

Estas son sólo unas pocas de las numerosas funcionalidades que nos ofrece el *framework*, en concreto en este módulo, ha resultado de gran utilidad el uso de directivas, ya que el módulo completo ha sido encapsulado en una directiva para permitir su utilización, tanto desde un *dashboard* propio, como desde un modal abierto desde un botón en las vistas de edición de los incidentes y eventos. En el caso de acceder desde una de estas dos entidades, la vista de selección de elemento base permanecería oculta y se tomaría como referencia el elemento con el que se está trabajando.

En la siguiente ilustración (ver Ilustración 13) se puede observar cómo se ha utilizado la directiva en el caso mencionado del modal.

```
<enriquecimiento-view ng-modal="true" element="element" ng-canedit="canEdit">
</enriquecimiento-view>
```

Ilustración 13. Ejemplo de inserción del módulo a través de su directiva.

En el proyecto se han utilizado algunas librerías de JavaScript que permiten, tanto mejorar la interfaz, como aprovecharnos de características ya desarrolladas, permitiendo así aumentar el número de funcionalidades sin necesitar un tiempo de desarrollo excesivo. Algunas de estas librerías y sus finalidades son las siguientes:

- **jQuery Clockpicker:** proporciona utilidades para trabajar con campos temporales, como son en el caso de este módulo los de las opciones de filtrado.
- **Bootstrap:** ofrece facilidades para obtener un diseño y una interfaz visualmente agradables para el usuario y con un gran nivel de usabilidad.
- **Bootstrap Tree View:** esta librería permite trabajar con estructuras en forma de árbol, en este módulo ha resultado de gran utilidad para representar en forma de árbol las diferentes capas ofrecidas para el filtrado, incluyendo la división en subniveles de estas.

6. Pruebas

La última fase del proceso de ingeniería software que se describe en esta memoria es la de pruebas. Este capítulo recoge las diferentes pruebas que se han realizado con relación al módulo desarrollado, para verificar su correcto funcionamiento y poder comprobar en versiones posteriores, que éste no se ve afectado.

6.1. Unitarias

Las pruebas unitarias tienen como finalidad verificar que cada uno de los módulos del sistema funcionan correctamente de manera aislada. A través de estas pruebas se verifica que la lógica de negocio se encuentra correctamente implementada y posteriormente, cuando se lleve a cabo alguna modificación, se vuelven a ejecutar para comprobar que la funcionalidad no se ha visto afectada.

En el proyecto en el que se incluye este módulo se optó por utilizar un *framework* de pruebas enfocado a aplicaciones desarrolladas en Node.js denominado Mocha.

Tras la ejecución de las diferentes pruebas unitarias configuradas para este módulo se obtiene una salida por consola con los resultados (ver Ilustración 14).

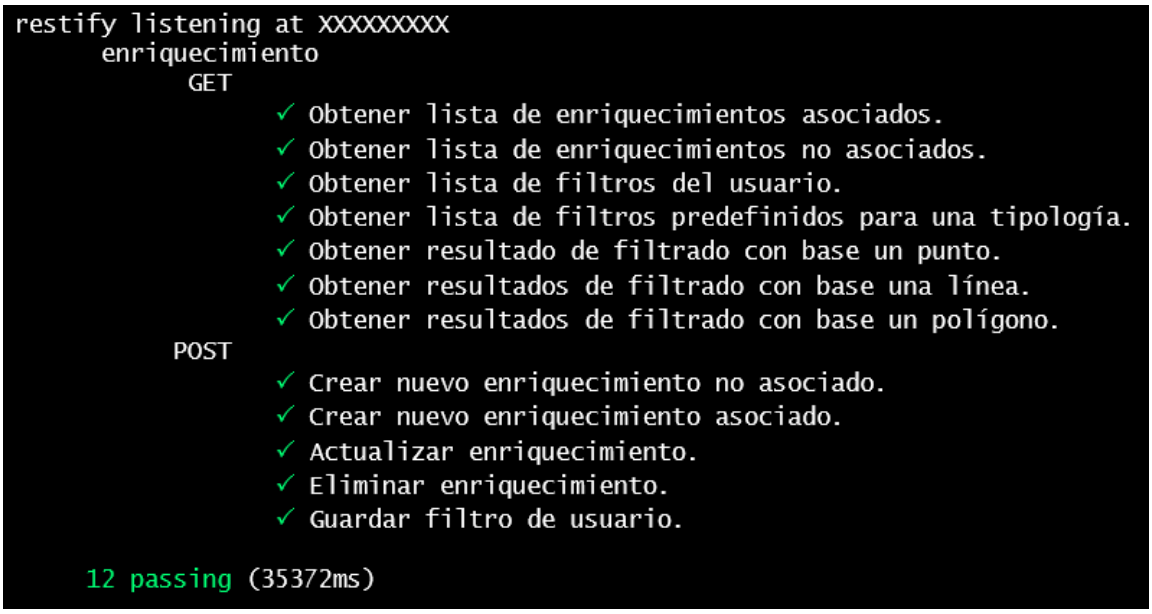

Ilustración 14. Resultado de la ejecución de las pruebas unitarias del módulo.

6.2. Integración

En segundo lugar, se encuentran las pruebas de integración cuyo objetivo es el de verificar que no existen errores en las interfaces e interacciones entre los diferentes componentes de la aplicación y entre la aplicación y los sistemas externos.

Se han realizado pruebas, que han ido aumentando a medida que se incluían más funcionalidades, para comprobar la correcta interacción entre los siguientes pares de capas:

- Capa de persistencia – Orion.
- Capa de persistencia – CrateDB.
- Capa de negocio – Capa de persistencia.
- Controladores – Capa de persistencia.

6.3. Sistema

Una vez se ha comprobado tanto el correcto funcionamiento de cada módulo de manera individual, como la correcta interacción entre estos, se procede a realizar las pruebas del conjunto completo para verificar que se cumple tanto con los requisitos funcionales como con los no funcionales. Para ello se ha llevado a cabo la ejecución de numerosas pruebas de forma manual dentro de una versión completa de la plataforma con todos los módulos ya integrados, replicando el entorno real.

Dentro del conjunto de pruebas de sistema se han llevado a cabo, por una parte, las comprobaciones relativas a la parte de seguridad y permisos de la aplicación y, por otra parte, verificaciones relativas a la estabilidad de la plataforma trabajando con este módulo. En relación con el módulo en el que se basa este proyecto, a través de la utilización de usuarios de prueba con diferentes perfiles asociados, se ha verificado si las acciones que les eran ofrecidas se correspondían con las esperadas y los datos a los que tenían acceso a través del módulo eran los adecuados.

En cuanto a la parte de comprobar la estabilidad del sistema, se ha hecho bastante hincapié en probar a realizar filtrados en los que el número de elementos a obtener de la base de datos del histórico sea elevado. Estas pruebas han tenido una gran importancia, ya que el sistema de CrateDB era uno de los más fatigados por la continua carga de datos y cuya estabilidad era clave para garantizar que no se pierden datos.

6.4. Aceptación

Por último, se llevan a cabo las pruebas de aceptación que consisten en verificar que el sistema cumple con los requisitos y objetivos definidos al inicio del proyecto y que, por tanto, se satisfacen las necesidades del cliente.

En este proyecto se dispone de un documento que incluye un conjunto de pruebas, en las que se definen procesos de uso de la plataforma paso por paso, definidas por el cliente y que son ejecutadas para verificar que se cumple con los criterios de aceptación. Las pruebas incluidas en dicho conjunto son en primer lugar ejecutadas por los propios desarrolladores y posteriormente, son ejecutadas por un equipo de QA para llevar a cabo una doble verificación del correcto funcionamiento de la plataforma.

7. Conclusiones y trabajos futuros

En el último capítulo se recogen las conclusiones a las que se ha llegado tras el desarrollo de este proyecto y se enumeran una serie de trabajos futuros que han ido surgiendo como ideas y que aún se desconoce si serán implementados en futuras versiones de la plataforma.

7.1. Conclusiones

En esta memoria se ha descrito el desarrollo de un nuevo módulo, integrado en una plataforma web de gestión de una ciudad inteligente, que ofrece la capacidad de poder analizar situaciones concretas en marcos espacio-temporales para las incidencias y eventos que suceden en la ciudad. El documento recoge los trabajos que se han llevado a cabo en cada una de las fases propias de la ingeniería de software, a excepción de la documentación, siguiendo una metodología de desarrollo iterativa e incremental, a través de la cual se dividió el trabajo a realizar en tres iteraciones.

Siguiendo el proceso indicado, se ha conseguido exitosamente tanto satisfacer los requisitos marcados con el cliente, como cumplir con los plazos temporales marcados para la entrega del módulo.

A nivel personal, durante los meses de desarrollo de este proyecto me he enfrentado a numerosos retos que me han ayudado a crecer tanto profesional como personalmente. Algunos de estos retos han sido, por ejemplo, trabajar con un gran número de tecnologías que desconocía y con las que he podido trabajar exitosamente gracias a las bases de conocimiento obtenidas durante mi etapa universitaria. Por otro lado, tratar directamente con el cliente, para exponerle los diferentes avances en el proyecto, me ha enriquecido en muchos aspectos de la comunicación humana tan importante en este negocio. También he descubierto lo verdaderamente importante que es la interrelación entre compañeros y la relevancia de apoyarse unos a otros, ya que cada uno tenemos nuestros puntos fuertes y nuestros puntos débiles.

7.2. Trabajos futuros

Durante el transcurso de una primera fase de la plataforma, en la que se incluye este desarrollo, se han ido planteando diferentes características que podrían considerarse muy útiles para ser implementadas en el futuro pero que aún no se han analizado ni acordado con el cliente. A continuación, se listan algunas de estas características:

- **Enriquecimientos automáticos realizados autónomamente por la plataforma:** incorporar la capacidad a la plataforma de detectar cuando existen conjuntos de datos alrededor del evento o incidencia que pueden resultar de utilidad para los operarios y guardar un enriquecimiento con dicha información de manera automática. Estos enriquecimientos se agruparían en otra pestaña, ya que sería un operario el encargado de incorporarlos a la lista de enriquecimientos asociados a la entidad, o bien, descartarlos.
- **Borrado selectivo de elementos no relevantes:** actualmente el módulo únicamente permite discriminar datos a través de los diferentes filtros, bien sea no seleccionando la capa en cuestión o ajustando los filtros espacio-temporales. Se plantea en un futuro permitir al usuario desvincular datos que no son relevantes y que por tanto no desea que persistan en el enriquecimiento del evento. Aun así, se guardaría como una nueva versión del enriquecimiento original para poder recuperar en cualquier momento el enriquecimiento que resultó del filtrado y conocer los elementos que fueron eliminados manualmente.
- **Actualización de las tecnologías utilizadas:** estudiar el impacto de migrar el proyecto a versiones más modernas de las tecnologías. Se plantea la opción de actualizar algunas de las tecnologías utilizadas en la plataforma, para ello habrá que estudiar tanto el coste como la necesidad de actualización de cada una de ellas. Existe la posibilidad de que alguna de las versiones del software utilizado ya no ofrezca una garantía de seguridad, y por tanto, esto puede suponer una brecha de seguridad para la aplicación.

8. Bibliografía

[1] Albino, Vito & Berardi, Umberto & Dangelico, Rosa. (2015). Smart Cities: Definitions, Dimensions, Performance, and Initiatives. Journal of Urban Technology. 22. 2015. DOI: 10.1080/10630732.2014.942092.

[2] Oracle. 2020. ¿Qué es big data? [Consulta: 21 enero 2020] Disponible en https://www.oracle.com/es/big-data/guide/what-is-big-data.html#link2

[3] Proyectos ágiles. 2020. Desarrollo iterativo e incremental. [Consulta: 21 enero 2020] Disponible en https://proyectosagiles.org/desarrollo-iterativo-incremental/

[4] Mozilla. 2019. Javascript. [Consulta: 21 enero 2020] Disponible en https://developer.mozilla.org/es/docs/Web/JavaScript

[5] AngularJS. 2020. What Is AngularJS? [Consulta: 31 julio 2019]. Disponible en https://docs.angularjs.org/guide/introduction

[6] IBM. 2011. ¿Simplemente qué es Node.js? [Consulta: 21 enero 2020]. Disponible en https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/index.html

[7] Mozilla. 2020. HTML: Hypertext Markup Language. [Consulta: 21 enero 2020] Disponible en https://developer.mozilla.org/en-US/docs/Web/HTML

[8] Mozilla. 2019. HTML5. [Consulta: 21 enero 2020] Disponible en https://developer.mozilla.org/es/docs/HTML/HTML5

[9] Mozilla. 2019. CSS3. [Consulta: 21 enero 2020] Disponible en https://developer.mozilla.org/es/docs/Archive/CSS3

[10] MongoDB. 2018. Introduction to MongoDB. [Consulta: 21 enero 2020]. Disponible en https://docs.mongodb.com/manual/introduction/

[11] CRATE.IO. 2020. CrateDB. [Consulta: 21 enero 2020]. Disponible en https://crate.io/products/cratedb/

[12] Atlassian. 2020. What is Git. [Consulta: 20 enero 2020]. Disponible en https://www.atlassian.com/git/tutorials/what-is-git

[13] JSON. 2020. Introducing JSON. [Consulta: 21 enero 2020] Disponible en https://www.json.org/

[14] Visual Studio. 2019. Why did we build Visual Studio Code? [Consulta: 21 enero 2020] Disponible en https://code.visualstudio.com/docs/editor/whyvscode

[15] Atlassian. 2013. Introducing Sourcetree for Windows – a free desktop client for Git. [Consulta: 21 enero 2020] Disponible en https://www.atlassian.com/blog/archives/introducing-sourcetree-git-client-microsoft-windows

[16] CenturyLink. 2014. What is Docker and When to Use It. [Consulta: 21 enero 2020] Disponible en https://www.ctl.io/developers/blog/post/what-is-docker-and-when-to-use-it/

[17] Apache. 2020. Apache NiFi Overview. [Consulta: 21 enero 2020] Disponible en https://nifi.apache.org/docs/nifi-docs/html/overview.html

[18] GeeksforGeeks. 2020. Introduction to Postman for API Development. [Consulta: 21 enero 2020] Disponible en https://www.geeksforgeeks.org/introduction-postman-api-development/

[19] SSH. 2020. PuTTY - World's Most Popular Free SSH Client. [Consulta: 21 enero 2020] Disponible en https://www.ssh.com/ssh/putty/

[20] WinSCP. 2020. Introducción. [Consulta: 21 enero 2020] Disponible en https://winscp.net/eng/docs/lang:es

[21] Rancher. 2020. Why Rancher? [Consulta: 21 enero 2020] Disponible en https://rancher.com/what-is-rancher/overview/

[22] Slack. 2020. ¿Qué es Slack? [Consulta: 21 enero 2020] Disponible en https://get.slack.help/hc/es/articles/115004071768--Qu%C3%A9-es-Slack-

[23] Office. 2020. Microsoft Project. [Consulta: 21 enero 2020] Disponible en https://products.office.com/es-es/project/project-and-portfolio-management-software

[24] Sommerville, I. 2011. Ingeniería del Software, 9ª Edición. Addison-Wesley. ISBN 978-607-32-0603-7.

[25] Chrome. 2020. MVC Architecture. [Consulta: 21 enero 2020] Disponible en https://developer.chrome.com/apps/app_frameworks

[26] IBM. 2020. What is machine learning? [Consulta: 8 febrero 2020] Disponible en https://www.ibm.com/topics/machine-learning

[27] FIWARE. 2020. Welcome to Orion Context Broker. [Consulta: 9 febrero 2020] Disponible en https://fiware-orion.readthedocs.io/en/master/index.html

[28] Express. 2020. Infraestructura web rápida, minimalista y flexible para Node.js. [Consulta: 9 febrero 2020] Disponible en https://expressjs.com/es/

[29] W3C. 2018. Web Content Accessibility Guidelines (WCAG) Overview. [Consulta: 9 febrero 2020] Disponible en https://www.w3.org/WAI/standards-guidelines/wcag/

[30] FIWARE. 2020. FIWARE-NGSI v2 Specification. [Consulta: 9 febrero 2020] Disponible en https://fiware.github.io/specifications/ngsiv2/stable/

[31] Mocha. 2020. Mocha simple, flexible, fun. [Consulta: 9 febrero 2020] Disponible en https://mochajs.org/